A Generalization of Tree Automata and Traversal of Trees(Fundamental Studies on Computational Complexity)

AUTHOR(S):

MORIYA, Etsuro

CITATION:

MORIYA, Etsuro. A Generalization of Tree Automata and Traversal of Trees(Fundamental Studies on Computational Complexity). 数理解析研究所講究録 1994, 876: 29-36

ISSUE DATE:

1994-06

URL:

http://hdl.handle.net/2433/84141

数理解析研究所講究録
第876巻 1994年 29-36

A Generalization of Tree Automata and Traversal of Trees

東京女子大学（文理） 守屋悦朗 (Etsuro MORIYA)

Abstract

Tree automata with memory, a generalization of ordinary tree automata, are introduced and their relation to context-free grammars with memory is studied. Relations between computation trees of tree automata with memory and derivation trees of context-free grammars with memory are established, which is a proper generalization of well-known Thatcher's theorem on tree automata. Some types of traversal of labeled trees are considered to characterize the languages generated by context-free grammars with memory.

1 Introduction

木オートマトンによって受理される labeled tree の集合と文脈自由文法の導出木の集合との間の関係は Thatcher の定理 [7] としてよく知られている。この定理は Guessarian [2] と Schimpf-Gallier [6] 等によってプッシュダウンメモリを持つ木オートマトンとインデックス文法の間の関係にまで拡張された。筆者は先にメモリ付きの文脈自由文法を考えることによってインデックス文法をスタック付きの文脈自由文法にまで拡張したが [3]、本論文ではプッシュダウン木オートマトンをスタックメモリ付きの木オートマトンにまで拡張することにより両者の間に同様な関係が成り立つことを示す。この結果より、スタックメモリを持つ木オートマトンはプッシュダウンメモリを持つ木オートマトンより真に受理能力が高いことが導かれる。

また、labeled tree に対する各種の traversal を考え、メモリ付き木オートマトンが受理する labeled tree を traverse して得られる言語とメモリ付き文脈自由文法との間の関係を考える。例えば、メモリ付き木オートマトンが受理する labeled trees を depth-first order や bottom-up order で traverse して得られる言語はメモリ付き文脈自由文法によって生成されることを示す。

Sections 2 and 3 are taken from [4] for the most part and therefore all the theorems in those sections are stated without proof. In Section 4 we consider various types of traversal of labeled trees and show that the language obtained by traversing the trees accepted by a stack tree automaton in the depth-first (or bottom-up) order is generated by a context-free grammar with stack.

2 Stack tree automata

Let Σ be an alphabet. A *tree domain* is a nonempty set $\mathbf{D}$ of strings over the set $\mathbf{N}$ of positive integers satisfying the following two conditions:

(i) For all d in $\mathbf{D}$, every prefix of d is also in $\mathbf{D}$.

(ii) For all d in $\mathbf{D}$ and every integer i in $\mathbf{N}$, if $d{\cdot}i$ is in $\mathbf{D}$, then for all j in $\mathbf{N}$ such that $1 \leq j \leq i$, $d{\cdot}j$ is also in $\mathbf{D}$.

A tree domain provides a scheme to identify uniquely each node of a rooted ordered tree, i.e., the root of the tree is denoted by the empty string λ and the ith child of a node d by $d{\cdot}i$. The number of children of d is denoted by $rank(d)$. A node d is a *leaf* if $rank(d) = 0$, and an *internal node* otherwise. Given an alphabet Σ, a Σ-*tree* (or *tree* for short) is a function $t : \mathbf{D} \to \Sigma$ for some tree domain $\mathbf{D}$ which we denote by $Dom(t)$. By $t/\mathbf{N}$ we denote the tree $\mathbf{D}/\mathbf{N} \to \Sigma$ obatined by pruning the entire leaves from t. Finally let $\mathbf{D} \bullet \mathbf{N} = \mathbf{D} \cup \{\ell{\cdot}i \mid \ell \text{ is a leaf of } \mathbf{D},\ i \in \mathbf{N}\}$.

Now we define tree automata with various types of worktape as a generalization of ordinary tree automata. We use the 'root-to-frontier' model instead of the 'frontier-to-root' model [7] and focus on tree automata with stack and their subclasses.

A *stack tree automaton* (*STA* for short) is a 7-tuple $M = (K, \Sigma, \Gamma, \delta, s_0, Z_0, F)$, where K is the finite set of *states*, Σ is the finite set of *labels* for trees, Γ is the finite set of *stack symbols* including the distinguished symbol Z_0, the *initial stack symbol*, $s_0 \in K$ is the *initial state*, $F \subset K$ is the set of *accepting states*, and δ is a mapping from $K \times (\Sigma \cup \{\lambda\}) \times \Gamma$ into the finite subsets of $\bigcup_n (K \times \Gamma^*)^n \cup \bigcup_n (K \times \{-1, 0, 1\})^n$. M is *λ-input-free* if the domain of δ is restricted to $K \times \Sigma \times \Gamma$. M is a *pushdown tree automaton* (*PDTA*) if $\delta(p, a, Z) \subset \bigcup_n (K \times \Gamma^*)^n$ for every p, a and Z, and M is a *finite tree automaton* (*FTA*) if $\Gamma = \{Z_0\}$ and $\delta(p, a, Z_0) \subset \bigcup_n (K \times \{Z_0\})^n$ for every p and a.

Given a tree $t : \mathbf{D} \to \Sigma$ as an input to $M = (K, \Sigma, \Gamma, \delta, s_0, Z_0, F)$, define the binary relation $\vdash_M$ on $\Gamma^* K \Gamma^*$-trees as follows. We may assume without loss of generality that $\Gamma \cap K = \emptyset$.

(i) The *initial tree* for M is the tree $init_M : \{\lambda\} \to \Gamma^* K \Gamma^*$ such that $init_M(\lambda) = s_0 Z_0$.

(ii) Let t_1 be a $\Gamma^* K \Gamma^*$-tree with tree domain $\mathbf{D}_1$ such that $\mathbf{D}_1 \subset \mathbf{D} \bullet \mathbf{N}$. Suppose $d \in \mathbf{D}$ is a leaf of t_1 such that $t(d) = a \in \Sigma$ and $t_1(d) = Z_1 \cdots p Z_i \cdots Z_k$, where p is in K and each of $Z_1, \ldots, Z_i, \ldots, Z_k$ in Γ. We write $t : t_1 \vdash_M t_2$ if one of the following four conditions is satisfied, where t_2 is a $\Gamma^* K \Gamma^*$-tree with tree domain $\mathbf{D}_2$ such that $t_2(d') = t_1(d')$ for every node d' other than those mentioned below.

(ii-1) $i = 1$, $\delta(p, \lambda, Z_1)$ contains $(q, \gamma) \in K \times \Gamma^*$, $\mathbf{D}_2 = \mathbf{D}_1$, and $t_2(d) = q\gamma Z_2 \cdots Z_k$.

(ii-2) $\delta(p, \lambda, Z_i)$ contains $(q, r) \in K \times \{-1, 0, 1\}$ such that $1 \le i + r \le k$, $\mathbf{D}_2 = \mathbf{D}_1$, and $t_2(d) = Z_1 \cdots Z_{i+r-1} q Z_{i+r} \cdots Z_k$.

(ii-3) d is an internal node of t with $rank(d) = n$, $i = 1$, $\delta(p, a, Z_1)$ contains $(q_1, \gamma_1; \ldots; q_n, \gamma_n) \in (K \times \Gamma^*)^n$, $\mathbf{D}_2 = \mathbf{D}_1 \cup \{d{\cdot}j \mid 1 \le j \le n\}$, and $t_2(d{\cdot}j) = q_j \gamma_j Z_2 \cdots Z_k$ for each j. In case d is a leaf of t, t_2 is defined similarly but unrelatedly to $rank(d)$ (in other words, n is arbitrary).

(ii-4) d is an internal node of t with $rank(d) = n$, $\delta(p, a, Z_i)$ contains $(q_1, r_1; \ldots; q_n, r_n) \in (K \times \{-1, 0, 1\})^n$ such that $1 \le i + r_j \le k$ for every j, $\mathbf{D}_2 = \mathbf{D}_1 \cup \{d{\cdot}j \mid 1 \le j \le n\}$, and $t_2(d{\cdot}j) = Z_1 \cdots Z_{i+r_j-1} q_j Z_{i+r_j} \cdots Z_k)$ for each j. In case d is a leaf of t, t_2 is defined similarly but unrelatedly to $rank(d)$.

In cases (ii-1) and (ii-2) M does not scan the node d to read a label there, i.e., M makes a *λ-move* at d. M is in *pushdown mode* if either case (ii-1) or case (ii-3) is applicable, and in *stack-reading mode* otherwise.

Let $\vdash_M^*$ be the reflexive transitive closure of $\vdash_M$. A $\Gamma^* K \Gamma^*$-tree t' is a *computation tree* of M on input t if $t : init_M \vdash_M^* t'$. t' is *acceptable* if $Dom(t'/\mathbf{N}) = Dom(t)$ and $t'(d)$ is in

$\Gamma^* F \Gamma^*$ for every leaf d of t'. A Σ-tree t is *accepted* (or *recognizable*) by M if there exists an acceptable computation tree of M on t. The set of trees accepted by M is denoted by $T(M)$, and the set of acceptable computation trees of M by $C(M)$.

3 Context-free grammars with memory

In the preceding paper [3] we introduced context-free grammars (CFGs) with various types of memory. In this paper we consider CFGs with stack, i.e., CFGs in which every nonterminal has a stack memory, exclusively among them. Formally, let $\underline{G} = (N, \Sigma, \underline{P}, S)$ be a CFG, where N is the set of nonterminals, Σ the set of terminals, $\underline{P}$ the set of context-free productions, and $S \in N$ the sentence symbol. Let Γ be a finite set disjoint from $N \cup \Sigma$, and let $\not\vdash$ and \$ be special symbols not in $N \cup \Gamma \cup \Sigma$. Elements of Γ are used as *stack symbols*. Then a *CFG with stack* (*CFGS* for short) is specified by a quintuple $G = (N, \Gamma, \Sigma, P, S)$, where P is the finite set of *productions* that have one of the following forms:

(I) $A \to \alpha$,
(II) $A \to Bf$,
(III) $Af \to B$,
(IV) $Af \to fB$, $Af \to Bf$ or $fA \to Bf$,

where A and B are in N, f in Γ, and α in $(N \cup \Sigma)^*$. G is *λ-free* if it has no production whose right side is λ. G is a *CFG with pushdown store* (*CFGP*) if it has no production of the form (IV). A CFGP is nothing other than an indexed grammar of Aho [1].

Intuitively speaking, each nonterminal of G has a stack of its own. An instantaneous content of the stack attached to nonterminal A is denoted by a string in $\not\vdash\Gamma^* A \Gamma^*\$$, the left-most symbol of which being the top of stack. The occurence of A denotes the read/write head's position on the stack. A *sentential form* of G is a string in $(\not\vdash\Gamma^* N \Gamma^*\$ \cup \Sigma)^*$ and is derived from the *initial sentential form*, $\not\vdash S\$$, as follows. Let β and γ be in $(\not\vdash\Gamma^* N \Gamma^*\$ \cup \Sigma)^*$, δ and ε in Γ^*, f in Γ, A, B in N, and $A_1, \ldots, A_k$ in $N \cup \Sigma$. Define the binary relation $\Rightarrow_G$ on sentential forms as follows.

(i) **Distribution** If $A \to A_1 A_2 \cdots A_k$ $(k \geq 0)$ is a production of type (I), then

$$\beta \not\vdash \delta A \varepsilon \$ \gamma \Rightarrow_G \beta \delta_1 A_1 \varepsilon_1 \delta_2 A_2 \varepsilon_2 \cdots \delta_k A_k \varepsilon_k \gamma,$$

where $\delta_i = \not\vdash\delta$ and $\varepsilon_i = \varepsilon\$$ if A_i is in N, and $\delta_i = \varepsilon_i = \lambda$ if A_i is in Σ. Note that $k = 0$ means $\delta_1 A_1 \varepsilon_1 \cdots \delta_k A_k \varepsilon_k = \lambda$.

(ii) **Pushdown** If $A \to Bf$ is a production of type (II), then $\beta \not\vdash A \varepsilon \$ \gamma \Rightarrow_G \beta \not\vdash B f \varepsilon \$ \gamma$.

(iii) **Pop Up** If $Af \to B$ is a production of type (III), then $\beta \not\vdash A f \varepsilon \$ \gamma \Rightarrow_G \beta \not\vdash B \varepsilon \$ \gamma$.

(iv) **Stack Reading** If $x \to y$ is a production of type (IV), then $\beta \not\vdash \delta x \varepsilon \$ \gamma \Rightarrow_G \beta \not\vdash \delta y \varepsilon \$ \gamma$.

Note that pushdown and pop up may be made only at the top of the stack. Let $\Rightarrow_G^*$ be the reflexive transitive closure of $\Rightarrow_G$. The language generated by G is denoted by $L(G)$, i.e.,

$$L(G) = \{w \in \Sigma^* \mid \not\vdash S\$ \Rightarrow_G^* w\}.$$

We denote by $\mathcal{L}_X$ the class of languages generated by grammars of type X. It is known [3] that $\mathcal{L}_{CFG} \subsetneq \mathcal{L}_{CFGP} \subsetneq \mathcal{L}_{CFGS}$.

For our purpose in this paper, we slightly modify the definitions of CFGSs and STAs. A CFGS is specified by a sextuple $G = (N, \Gamma, \Sigma, P, S, Z_0)$, where N, Γ, Σ and S are as before, Z_0 is a distinguished symbol of Γ called the *initial stack symbol*, and P may contain productions of the forms (II), (III) or (IV), as well as productions of the form

(IV′) $fAg \to Bfg$

and

(V) $Af \to B\theta$,

where A and B are in N, f and g in Γ, and θ in Γ^*. However, productions of the form (I) are restricted such that $\alpha \in N^* \cup \Sigma$. The initial sentential form of G is $\mathfrak{c}SZ_0\$$. Production (IV′) has exactly the same effect as applying two productions $Ag \to Cg$ and $fC \to Bf$ in this order, while production (V) has the composite effect of applying a type (III) production first and then a sequence of type (II) productions, i.e., it can be applied only in the pushdown mode to drive $\beta\mathfrak{c}B\theta\varepsilon\$\gamma$ from $\beta\mathfrak{c}Af\varepsilon\$\gamma$. Particularly production $Af \to Bf$ may be used not only as a type (V) production but also as a type (IV) production.

A STA $(K, \Sigma, \Gamma, \delta, s_0, Z_0, F)$ is restricted such that $\delta(p, a, Z) \subset \bigcup_{n\geq 2}((K-F)\times\{0\})^n \cup K \times (\Gamma^* \cup \{-1, 0, 1\})$ and such that $\delta(p, \lambda, Z) \subset K \times (\Gamma^* \cup \{-1, 0, 1\})$ for every $p \in K$, $a \in \Sigma$ and $Z \in \Gamma$.

It can be shown that the above mentioned modifications neither increase nor decrease the power of the devices, and do not change essentially the structure of them, either.

Derivation trees for $G = (N, \Gamma, \Sigma, P, S, Z_0)$ are those $\Gamma^*(N \cup \Sigma \cup \{\lambda\})\Gamma^*$-trees which are defined recursively as follows.

(i) A single-node tree $init_G : \{\lambda\} \to \{SZ_0\}$ is a derivation tree called the *initial derivation tree* for G .

(ii) Suppose $t : \mathbf{D} \to \Gamma^*(N \cup \Sigma \cup \{\lambda\})\Gamma^*$ is a derivation tree with a leaf ℓ. Then the tree $t' : \mathbf{D}' \to \Gamma^*(N \cup \Sigma \cup \{\lambda\})\Gamma^*$ defined below is a derivation tree, where $t'(d) = t(d)$ for every d in $\mathbf{D}$. In this case, we write $t \Rightarrow_G t'$.

(ii-1) If $t(\ell) = \delta A\varepsilon$ and $A \to A_1 \cdots A_k$ ($k \geq 1$, each $A_i \in N \cup \Sigma$) is a production of type (I), then $\mathbf{D}' = \mathbf{D} \cup \{\ell{\cdot}i \mid 1 \leq i \leq k\}$ and $t'(\ell{\cdot}i) = \delta A_i\varepsilon$.

(ii-2) If $t(\ell) = \delta A\varepsilon$ and $A \to \lambda$ is a production, then $\mathbf{D}' = \mathbf{D} \cup \{\ell{\cdot}1\}$ and $t'(\ell{\cdot}1) = \delta\varepsilon$.

(ii-3) If $t(\ell) = x\varepsilon$ and $x \to y$ is a production of type (II), (III) or (V), then $\mathbf{D}' = \mathbf{D} \cup \{\ell{\cdot}1\}$ and $t'(\ell{\cdot}1) = y\varepsilon$.

(ii-4) If $t(\ell) = \delta x\varepsilon$ and $x \to y$ is a production of type (IV) or (IV′), then $\mathbf{D}' = \mathbf{D} \cup \{\ell{\cdot}1\}$ and $t'(\ell{\cdot}1) = \delta y\varepsilon$.

An *acceptable* derivation tree is a derivation tree whose leaves each is labeled with an element of $\Gamma^*(\Sigma \cup \{\lambda\})\Gamma^*$. The set of acceptable derivation trees for G is denoted by $T(G)$. Let $t : \mathbf{D} \to \Gamma^*(N \cup \Sigma \cup \{\lambda\})\Gamma^*$ be a derivation tree with the leaves $\ell_1, \ldots, \ell_n$ from left to right. Then the string $t(\ell_1) \cdots t(\ell_n)$ is denoted by $yield(t)$. If T is a set of trees, then let $yield(T) = \{yield(t) \mid t \in T\}$.

Throughout the paper, let π be the homomorphism from $(\Gamma \cup N \cup \Sigma)^*$ into $(N \cup \Sigma)^*$ which maps each element of Γ into λ and each element of $N \cup \Sigma$ into itself. Then

$L(G) = \pi(yield(T(G)))$.

Let $\varphi : \Sigma \to \Delta$ be a mapping and $t : \mathbf{D} \to \Sigma$ a tree. Then by $\varphi(t)$ we denote the tree $t' : \mathbf{D} \to \Delta$ defined by $t'(d) = \varphi(t(d))$ for each d in $\mathbf{D}$. φ is called a *relabeling* of t. For a set T of Σ-trees, let $\varphi(T) = \{\varphi(t) \mid t \text{ in } T\}$. The following theorems are proved in [4].

Theorem 3.1 *For each λ-free CFGS (CFGP, CFG, respectively) G, there exists a λ-input-free STA (PDTA, FTA, respectively) M and a relabeling φ such that $T(G) = \varphi(C(M))$.*

Corollary 3.1 *For each λ-free CFGS (CFGP, CFG, respectively) G, $\pi(T(G))$ is recognizable by a λ-input-free STA (PDTA, FTA, respectively).*

Corollary 3.2 *For each λ-free CFGS (CFGP, CFG, respectively) G, there exists a λ-input-free STA (PDTA, FTA, respectively) M such that $L(G) = yield(T(M))$.*

Theorem 3.2 *For each λ-input-free STA (PDTA, FTA, respectively) M, there exists a λ-free CFGS (CFGP, CFG, respectively) G and a relabeling φ such that $C(M) = \varphi(T(G)/\mathbf{N})$.*

Corollary 3.3 *For each λ-input-free STA (PDTA, FTA, respectively) M, there exists a λ-free CFGS (CFGP, CFG, respectively) G such that $yield(T(M)) = L(G)$.*

From Corollaries 3.2 and 3.3, we have the following theorem which is a generalization of the corresponding results for context-free languages [7] and for indexed languages [2] [6].

Theorem 3.3 *The following two statements are equivalent.*
(1) L is generated by a λ-free CFGS (CFGP, CFG, respectively).
(2) $L = yield(T(M))$ for some λ-input-free STA (PDTA, FTA, respectively).

Since it is known [3] that $\mathcal{L}_{CFG} \subsetneq \mathcal{L}_{CFGP} \subsetneq \mathcal{L}_{\lambda-free\ CFGS}$, the next theorem follows from Theorem 3.3, where $\mathcal{T}_X$ denote the class of the sets of trees accepted by tree automata of type X.

Theorem 3.4 $\mathcal{T}_{FTA} \subsetneq \mathcal{T}_{PDTA} \subsetneq \mathcal{T}_{STA}$.

4 Traversing trees

In this section we consider traversal of trees according to the depth-first and the bottom-up orders. Let $t : \mathbf{D} \to \Sigma$ be a tree. We identify the sequence $d_1, \ldots, d_n$ of nodes constituting a traversal with the string $t(d_1) \cdots t(d_n)$ of labels. Let $<_{\mathbf{D}}$ be the lexicographic order of $\mathbf{D}$, i.e., $\alpha <_{\mathbf{D}} \beta$ if and only if either $\alpha = \lambda$ and $\beta \neq \lambda$, or $\alpha = \gamma i \alpha'$ and $\beta = \gamma j \beta'$ for some $\alpha', \beta' \in \mathbf{N}^*$ and $i, j \in \mathbf{N}$ such that $i < j$. The *depth-first traversal* (pre-order traversal) of t, denoted by $depth\text{-}first(t)$, is the traversal according to the order $<_{\mathbf{D}}$. The *bottom-up traversal* (post-order traversal), *bottom-up*(t), is the traversal according to the reverse order of $<_{\mathbf{D}}$. For a set T of trees, let $trav(T) = \{trav(t) \mid t \in T\}$, where $trav$ is one of the traversals defined above.

Theorem 4.1 *Let M be a λ-free STA (PDTA, FTA, respectively). Then there exists a λ-free CFGS (CFGP, CFG, respectively) G such that $L(G) = depth\text{-}first(T(M))$.*

Proof. Let $M = (K, \Sigma, \Gamma, \delta, s_0, Z_0, F)$. Let $p, \bar{a}, [a,p]$ and $[a, q_1, \cdots, q_k]$ be nonterminals of G for each $p, q_1, \cdots, q_k \in K$ and $a \in \Sigma$, where $k \leq \max\{k \mid (q_1, 0; \cdots; q_k, 0) \in \delta(p, a, Z),\ p \in K,\ a \in \Sigma,\ Z \in \Gamma\}$, and let Z and $\bar{Z}$ be stack symbols of G for each $Z \in \Gamma$. Let s_0 and Z_0 be the sentence symbol and the initial stack symbol of G, respectively.

Before definig the set P of productions for G, it would be helpful for understanding to know the basic idea of the proof. Suppose $t : init_M \vdash_M^* t'$ with $t' \in C(M)$. Let s' be the tree obtainded from t' by performing the following sequence of operations on every node d of $t'/\mathbf{N}$. Note that $Dom(t) = Dom(t'/\mathbf{N})$.

1. Create a new node d_1 to be the unique child of d.
2. **if** d is an internal node of $t'/\mathbf{N}$ **then begin**
   2.1. Make the children of d in t' be the children of d_1 in s', preserving their order.
   2.2. Add a new node d_2 as the leftmost child of d_1.
   2.3. Create another new node d_3 as the unique child of d_2.
   2.4. Suppose $t'(d) = \alpha p \beta$ for some $p \in K$ and $\alpha, beta \in \Gamma^*$.
   2.4.1. Label d_3 with $\alpha t(d) \beta$.
   2.4.2. Label d_2 with $\alpha \bar{a} \beta$.
   2.4.3. Label the remaining children of d_1 with the labels of the corresponding children
   of d in t'.
   end
3. **else if** d is a leaf of $t'/\mathbf{N}$ **then begin**
   3.1. Create a new node d_3 to be the unique child of d_1.
   3.2. Label d_3 with $\alpha a \beta$ if $t'(d) = \alpha p \beta$ for some $p \in F$.
   end

It is important to observe that $depth\text{-}first(t) = h(yield(s'))$, where h is the homomorphism which erases every stack symbol and maps each terminal symbol to itself. This is the case because the order of nodes obtained by traversing t in the depth-first order is equivalent to the order of the nodes obtained by traversing $t'/\mathbf{N}$ in the depth-first order, and because each node, say d, of t' has the unique d_3 for which $h(s'(d_3)) = t(d)$, as its leftmost descendant leaf in s'.

Now we are ready to define the set P of productions for G. Let $p, q, q_1, \cdots, q_k \in K$, $a \in \Sigma$ and $Z \in \Gamma$.

(a) If $\delta(p, a, Z)$ contains $(q_1, 0; \ldots; q_k, 0)$ for some $k \geq 2$, then P contains $pZ \to [a, q_1, \cdots, q_k]Z$. Note that this production is of type (IV) and thus can be applied in the stack reading mode.

Nonterminals of the form $[a, q_1, \cdots, q_k]$ or $[a, q]$ are introduced in order for G to remember a label a which M has scanned on the input tree and a(n) (ordered set of) state(s)

q $(q_1, \cdots, q_k)$ which M is supposed to enter in the subsequent move. After that, G produces, on its derivation tree, $\bar{a}$ (and then its unique child a) and q $(q_1, \cdots, q_k)$ surrounded by strings of stack symbols as the labels of child nodes of the node corresponding to the node M has just scanned on the input tree. These nodes, except for the leftmost one which has $\bar{a}$ as its label, have the same label that the corresponding nodes on a computation tree of M have.

(b-1) If $\delta(p, a, Z)$ contains (q, γ) with $\gamma \neq Z$, then P contains $pZ \to [a, q]\gamma$ and $p\bar{Z} \to [a, q]\gamma$.

If $\gamma = Z$ then $pZ \to [a, q]\gamma$ can be regarded not only as a type (V) production but also as a type (IV) production, whereas it should not be treated as a type (IV) production because the corresponding move in M is to be made in the pushdown mode. The stack symbols in $\bar{\Gamma}$ are introduced in order for G to be able to recognize this difference. Except for this point, $\bar{Z}$ is treated in G as if it were Z.

(b-2) If $\delta(p, a, Z)$ contains (q, Z), then P contains $pZ \to [a, q]\bar{Z}$ and $p\bar{Z} \to [a, q]Z$. Note that this production is of type (V) and thus can be applied only in the pushdown mode.

(c) If $\delta(p, a, Z)$ contains $(q, -1)$, then P contains $YpZ \to [a, q]YZ$ and $Yp\bar{Z} \to [a, q]Y\bar{Z}$ for each Y in $\Gamma \cup \bar{\Gamma}$.

(d) If $\delta(p, a, Z)$ contains $(q, 0)$, then P contains $pZ \to [a, q]Z$ and $p\bar{Z} \to [a, q]\bar{Z}$.

(e) If $\delta(p, a, Z)$ contains $(q, 1)$, then P contains $pZ \to Z[a, q]$ and $p\bar{Z} \to \bar{Z}[a, q]$.

(f) For each $q, q_1, \cdots, q_k \in K$ and $a \in \Sigma$, P contains $[a, q] \to \bar{a}q$ and $[a, q_1, \cdots, q_k] \to \bar{a}q_1 \cdots q_k$. The role of nonterminal $\bar{a}$ in (f) and (g) is to adjust the form of productions. Thus these productions are of type (I).

(g) For each $q, q_1, \cdots, q_k \in F$ and $a \in \Sigma$, P contains $[a, q] \to a$ and $[a, q_1, \cdots, q_k] \to a$.

Now it is easily seen that $L(G) = h(yield(T(G))) = depth\text{-}first(T(M))$.

In the above proof, replace (f) by (f') below. Then we have the corresponding result for the bottom-up traversal.

(f') For each $q, q_1, \cdots, q_k \in K$ and $a \in \Sigma$, P contains $[a, q] \to q\bar{a}$ and $[a, q_1, \cdots, q_k] \to q_1 \cdots q_k\bar{a}$.

Theorem 4.2 *Let M be a λ-free STS (PDTA, FTA, respectively). Then there esists a λ-free CFGS (CFGP, CFG, respectively) such that $L(G)$ =bottom-up$(T(M))$.*

Acknowledgement

The author (A) is very grateful to Prof. H. Machida (B) for his efforts to organize the comfortable conference where he was permitted not to give a presentation but to publish this paper in the proceedings. As A is from Yamanashi prefecture, he wishes to express his sincere appreciation to B in 甲州弁 dialect, "いつものこんだけど、えらいこんずら。わるいね。"

References

[1] Aho, A. V., Indexed grammars— an extension of context-free grammars, *J. Assoc. Comput. Mach.* **15** (1968), 647-671.

[2] Guessarian,I., Pushdown tree automata, *Math. Systems Theory* **16** (1983), 237-263.

[3]. Moriya, E., Context-free grammars with memory, *IEICE Trans. Inf. Syst.* **E-75-D** (1992), 847-851.

[4] ——, Stack tree automata and their relation to context-free grammars with memory, submitted for publication.

[5] ——, Generalized tree automata over partially labeled tree, in preparation.

[6] Schimpf, K. M. and J. H. Gallier, Tree pushdown automata, *J. Comput. Sys. Sci.* **30** (1985), 25-40.

[7] Thatcher, J. W., Tree automata: an informal survey, in: A. V. Aho (ed.), *Currents in the Theory of Computing* (Prentice-Hall, Englewood Cliffs, N. J., 1973).